H.J. Thim Trust's

Theem College of Engineering,
BoisarChilhar Road, Boisar (E), Tal - Palghar

ACADEMIC YEAR: 2024-25 CLASS: B.E Comp SEM: VIII

NAME: Siddhi P. Kattkar

ROLL NO: 15 BATCH: B1

SUBJECT: Deep Learning

EXPERIMENT /ASSIGNMENT NO: 1

AIM: Assignment No. 1

DATE OF PERFORMANCE: / /25

DATE OF SUBMISSION: / /25

PARAMETER	C	P	A	Total	Sign. With Date
MARKS OBTAINED	2	2	1	5	
MAX.MARKS	2	2	1	5	

DL

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Assignment No. 1

Date :

1. Explain with an example McCulloch-Pitts neuron model.

> The McCulloch-Pitts neuron model is one of the earliest mathematical models of a neuron. It represents a neuron as a threshold unit where the inputs are binary are the neuron fire if the weighted sum of the inputs exceeds a threshold.

Steps in a model:
1. Input are multiplied.
2. The weighted inputs
3. The Sum is compared

Example

x_1	x_2	Output
0	0	0
0	1	0
1	0	0
1	1	1

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

2\. Explain perception learning with the help of example.

→ Perception learning is a supervised learning algorithm for binary classifiers. The perception adjusts weights iteratively to minimise errors.

Steps :-

1. Initialise weights randomly.
2. Comput output
3. Update weights if weight is incorrect

$$\omega_1 = \omega_1 + D\omega_1$$

$$D\omega_1 = n(d-y)$$

Hence n is learning rate,
d is desired output.

Example :-

Classifying points (0,0), (0,1), (1,0) & (1,1) for an AND gate

3. Prove the following identities :-

→

1. For unipolar continuous activation function f(net) $= 1 / (1 + e^{-net})$

$$f'(net) = f(net)\,(1 - f(net))$$

Proof :- let $f(net) = y = 1 / (1 + e^{-net})$

Differentiating y :-

$$f'(net) = y(1-y)$$

2. For bipolar continuous activation function $f(net) = (1 - e^{-net}) / (1 + e^{-net})$:-

$$f'(net) = (1 - f^2(net)) / 2$$

Proof :- This involves differentiating the tan function.

Date :

4. Explain SCPTA.

⇒ The SCPTA is an iterative process for training perceptions with continuous activation functions.

Steps :-

1. Initialize weights.
2. Compute the weighted sum net $= \Sigma(w_i \cdot x_i)$
3. Apply continuous activation function
4. Update weights using gradient descent
   $w_i = w_i + n\,(d-y)\,x_i)$

5. Explain with example linearly and non-linearly seperable pattern into classification.

Example :- Point (0,0), (0,1), (1,0), (1,1) for OR gate.

Ex. Non-linearly seperable :- Classes can't be seperated by line.

Example XOR gate for output 1 (0,1) & (1,0) and 0 otherwise.

H. J. Thim Trust's

THEEM COLLEGE OF ENGINEERING

Date : ______

6. Find the output of neuron for given input & weights.

→

1. IP vector $x = [5, 6, 7, 8]$

$w = [1, 2, 3, 4]$

$net = \Sigma w_i x_i$]

$f(net) = 2net + 1$

$f(net) = \frac{1}{1+e^{-net}}$

$net = 5 + 12 + 21 + 32 = 70$

$f(net) = 2 \cdot net + 1$

$= 141$

binary sigmoid)

$f(net) = 1/1+e^{-70}$

$= 1$

The output is 1.

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

7. Neuron with 8 inputs where
$\omega = [0.1, 0.2, -0.1, 0.2]$
$x = [0.8, 0.6, 0.4, 0.9)$

Find output of neuron.

→ net $\Sigma(\omega_i, x_i)$
net $= 0.08 + 0.12 - 0.04 + 0.14$
net $= 0.3$

binary sigmoid =
$f(net) = \frac{1}{1+e^{-net}}$

$= \frac{1}{1+e^{-0.3}}$ $= \frac{1}{1+0.7408}$
$= 0.574$

Output of the neuron is approx 0.574

8. A single neuron network using $f(net) = Sgn(net)$
Training pairs.

$X_1 = [1, -2, 3, -1]^T$ $\quad d_1 = -1$

$X2 = [0, -2, 0, -1]^T$ $\quad d_2 = 1$

$X3 = [-2, 0, -3, 1]^T$ $\quad d_3 = -1$

The final weight are $[3, 2, 61]^T$

· Backtracking

$\omega_3, \omega_2, \omega_1$

$net_3 = \omega_n + 1 = \omega_n + D\omega, \; D\omega = (d-y)X$

$D\omega_3 = (d_3 - y_3) X_3$

$y_3 = 13 - 6 + 0 - 18 - 1 = -25$

$y_3 = Sgn(-25)$

$= -1$

$D\omega_2 - (d_2 + y_2) X_2$

$net_2 = \omega_3^T X_2 \; [3, 2, 6, 1]^T \cdot [0, -2, 0, -1]^T$

$net_2 = 0 - 4 + 0 - 1 = -5$

Weight Summary

$\omega_4 = [3, 2, 6, 1]$

$\omega_3 = [32, 6, 1]^T$

$\omega_2 = [3, 6, 6, 3]^T$

$\omega_1 = [5, 2, 12, 1]^T$

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date : ____________

9. Determine the weight after 4 step of train for a perception learning rule.

Initial weight $= w = [0,0,0]^T$

$x_1 = [2, 2, 1]^T$ $d_1 = 1$

$x_2 = [2, -2, 1]^T$ $d_2 = -1$

$x_3 = [2, 2, 1]^T$ $d_3 = 1$

$y_1 = \text{Sgn}(net_1) = 0$

$net_2 = 1$

$Dw_2 = [-2, 6, -1]$

$X_3 =$

$y_3 = \text{sgn}(15) = 1$

$y_3 = d_3$

$w_3 = w_2 = (-2, 6, -1)$

Final weight $= [-2, 6, -1]^T$

$w_1 = w_2 = w_3 = w_4$

10. Explain biological neuron with a digram.

→ A biological neuron serves as a foundation for artificial newon in DL.

Key component

Dendrite

Cell body

Axon

Synapse.

Diggram:-

$\boxed{\text{Input signal}}$

↓

$\boxed{\text{Dendrites}}$ → $\boxed{\text{Cell body}}$

↓

$\boxed{\text{Axon}}$ → Synapse

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date : ____________

11. Show how NAND logic can be realized using Mc Culloch Pitts model.

→ Configure weight & threashold for neuron

A	B	O/P
0	0	1
0	1	1
1	0	1
1	1	0

12. Explain McCulloch Pitts Model & implement OR.

→ The MP model implifies biological neuron in mathematical model for decision making.

OR Gate:-

X1	X2	OR O/P
0	0	0
0	1	1
1	0	1
1	1	1

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

13. State and explain different activation function.

→

1. Step function :-
   - O/P is binary (0,1)

   $f(net) = \begin{cases} 1, & \text{if } net \geq 0 \\ 0, & \text{otherwise} \end{cases}$

2. Sigmoid function :-
   - Smooth curve output between 0 & 1

   $f(net) = 1 / 1 + e^{-net}$

3. Tanh function :-
   - O/P between -1 & 1

   $f(net) = \tan h\,(net)$

4. Relu function

   Output is max (0, net)

15

$\frac{05}{05}$

22/2/25

H.J. Thim Trust's

Theem College of Engineering,
BoisarChilhar Road, Boisar (E), Tal - Palghar

ACADEMIC YEAR: 2024-25 **CLASS:** BE COMP **SEM:** VIII

NAME: Siddhi P. Katkar

ROLL NO: 15 **BATCH:** B1

SUBJECT: Deep Learning

ASSIGNMENT NO: 2

DATE OF PERFORMANCE: ____/____/__

DATE OF SUBMISSION: ____/____/____

PARAMETER	C	P	A	Total	Sign. With Date
MARKS OBTAINED	2	2	1	5	
MAX.MARKS	2	2	1	5	

DL

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Assignment No. 2

Date :

1] Explain multilayer feed forward network architecture.
→ A multilayer feedback network consists of three key layers

a. Input layers :
- The first layer in the network layer.
- Receives raw input data (e.g. images, text or numbers)
- does not perform computation : simply passes inputs to the next layer.

b. Hidden layer(s):
- One or more layers between the input & output layers.
- more hidden layers = deeper neural network (Deep Neural Network - DNN)
- Helps the network learn complex patterns in the data

c. Output layer :
- The final layer in the network.
- Produces the final predicted output.
- Uses an activation function

2. Working
Step 1 : Forward Propagation

- Each neuron receives input from the previous layer
- each input is multiplied by its corresponding weight.
- The weighted sum is computed.

$$net = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b$$

where w is the weight, x is the input and b is the bias.

The activation function f(net) is applied to introduce non-linearity.

- The output is passed to the next layer until it reaches the final output layer.

Step 2: Error calculation

The difference between actual output and predicted output is measured using a loss function (e.g mean squared error, cross-entropy loss)

Step 3: Back Propagation & weight update

- The error is propogated backward through the network using back propogation.
- Gradient descent is used to update the weights.
- The process continue for multiple epochs until the error is minimized.

3 Example.
Consider a 3 layer network for classifying images for handwritten digits.
Input layer : 784 neurons
Hidden layer : 128 neurons
Output layer : 10 neurons
$x \rightarrow w1 + b1 + f(RELU) \rightarrow w2 + b2 \rightarrow f(softmax) \rightarrow Y_{predicted}$

- Advantages :-
- Can model linear relationship using activation function
- Works well for classification & regression tasks.
- Can extract high-level features from raw data.

- Disadvantages :-
- Requires large amount of data & high computational power.
- Prone to overfitting if too many hidden layers are used.
- Training can be slow due to backpropogation.

2] Explain important learning ~~lr~~ factor that affects learning performance

→ Learning Factor :-

There are many factors that affects the neural network learning.

Some factors are :-

1. Initial Weight :-

Usually the weights of the network are initialized to small random values.

The initial values of weights strongly affect the ultimate solution.

If all weights start out with equal weight values and if the solution requires that unequal weights developed, the network not be trained properly.

2. Choice of Activation function :-

The choice of Activation function is a critical part of neural network design.

. The choice of activation function in the hidden layer will control how well the network model learns training dataset

. The choice of activation function in the output layer will define the type of predictions the models can make.

3. Learning Constant -
The effectiveness and convergence of learning algorithm depends significantly on the value of learning constant.
However the optimum value of n depends on the problem solved & there is no single learning constant value suitable for different training cases.

4. Network Architecture -
One of the most important attributes if a layed neural network design is choosing the Architecture.
The number of input nodes is simply determined by the dimension or size of the input vector to be classified.
The input vector size usually corresponds to the total no. of distinct features of the input patterns

5. Momentum :
The purpose of the momentum method is to accelerate the convergence of the EBPT (error Back Propogation training)

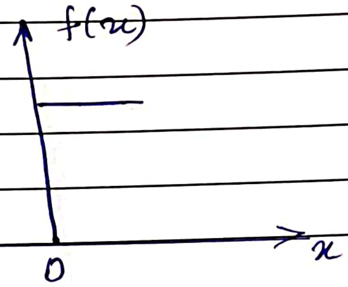
6. Loss Function :
Deep learning neural networks are trained using the stochastic gradient descent optimization algorithm.

As part of the optimization algorithm the error for the current state of the model must be essential repeated.

7. Epochs and Batch Size:-
The Batch size refers to the number of samples that must be processed before the model is updated number of epochs refers to the total no. of times the training database has been transvered to trained.

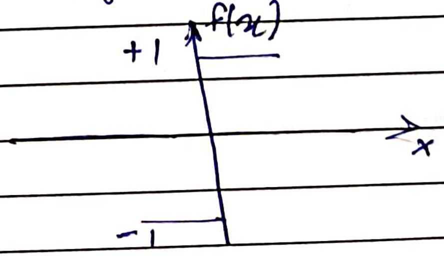
2. Unipolar Binary or Binary Step function :-
Binary step function depends on a threshold value that decides whether a neuron should be activated or not.
Mathematically it can be represented as,

$$o = f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$



- Uses :- This function is used for binary classification problem.

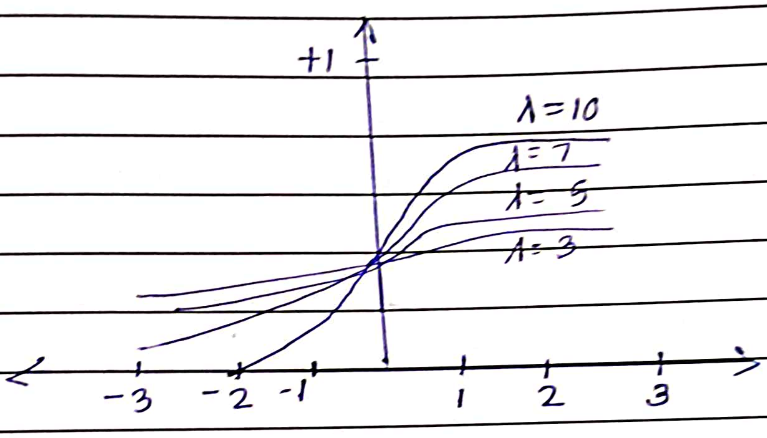
3. Bipolar Binary
It is similar to unipolar binary function. The only difference is its produces out as either -1 or 1.



$$f(x) = \mathrm{sign}(x) = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \end{cases}$$

Uses : This function is also used for binary classification

4. Sigmoid Activation function:
Sigmoid are called as logistic function.
It takes a real-valued number and "squashes" it into the range between 0 & 1.

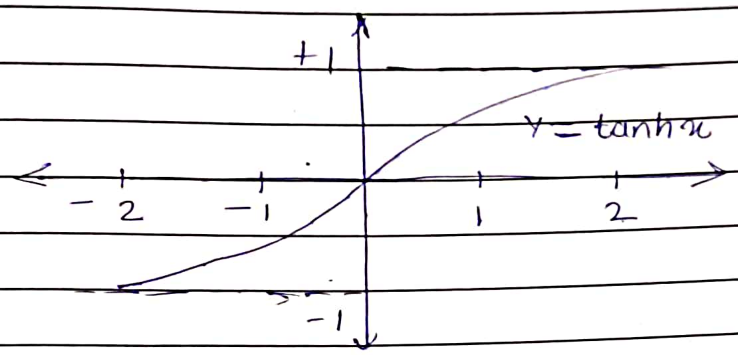


$$f(net) = \frac{1}{1+\exp(-\lambda\, net)}$$

Uses : This function is used in output layer.

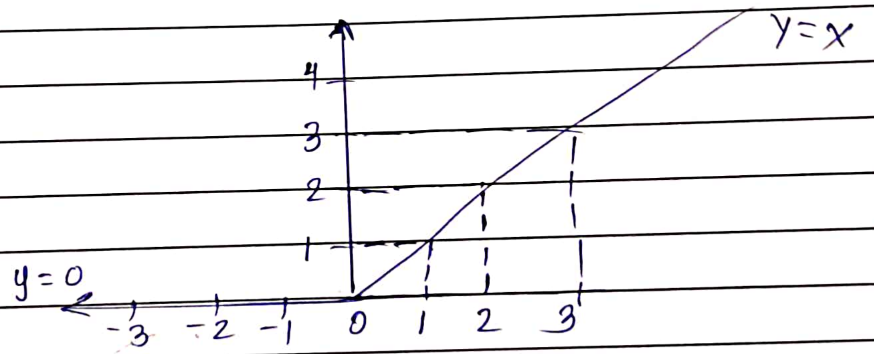
5. Tanh function
It takes a real-valued no. and "squashes" it into range between -1 & 1.
Unlike Sigmoid the output is zero-centered it is therefore preferred than sigmoid.

$$f(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$$



Uses: Hidden layer, especially in RNNs.

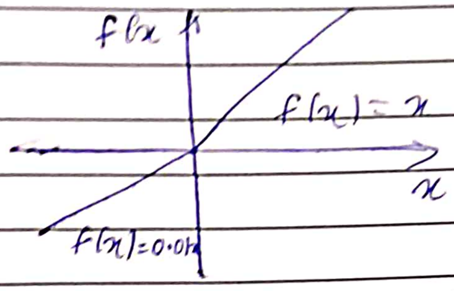
6. ReLU function
Takes real valued number & threshold it a zero.
$f(x) = \max(0, x)$



Uses: Hidden layers of deep networks, CNNs & feedforward network.

7. Leaky ReLU function.

Small slope for negative instead of a function slope



$$f(x) = \begin{cases} x & x>0 \\ 0.01x & x \le 0 \end{cases}$$

Uses: Hidden layer of deep networks, often in CNNs

4] Explain different loss functions along with its application.

→ 1. Regression loss function.
Used when predicting continuous values.

a) Application: Mean Squared Error
Used in regression problem like stock price prediction. $MSE = \frac{1}{n} \sum (y_i - \hat{y}_i)^2$

b) Mean Absolute error.
$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$

Used where robustness to outliers is needed.

2. Classification Loss Functions

Used when predicting discrete categories.
a) Binary Cross Entropy.
$\frac{1}{n} \sum^{n} [y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}_i)]$
Used for binary classification

b) Hinge Loss
$Hinge = \sum_{i=1}^{n-1} max(0, 1 - y_i \hat{y}_i)$

Use for SVMs and DL requiring margin based classification.

5] Explain different types of Biases?

→ In deep learning bias refers to systematic errors in model predictions due to incorrect assumptions in the learning process. Biases can come from data, model design, human decision.

1. Algorithm Bias

Arises due to design of learning algorithm.
E.g. A model trained on imbalanced data may favor the dominant data.

2. Data Bias

Occurs when training dataset does not represent the real world distribution.
Eg. Facial recognition model trained on mostly light-skinned faces may perform poorly on dark skinned faces.

3. Selection Bias

Happens when the data use to train the model is not randomly selected.
E.g Medical diagnosis model trained on data from urban hospitals may not generalize well to rural population.

4. Confirmation Bias
Happens when models reinforce existing beliefs or patterns in data.
Eg Recommendation system that suggests similar content repeatedly, limiting diversity.

5. Social Bias
Results from human biases present in the data.
Eg Hiring model trained on past recruitment data may prefer male candidates if historical hiring favoured men

6. Bias Variance Tradeoff.
Underfitting – Model is too simple & fails to capture patte[rns]
Overfitting – Model learns noise from training data & fails to generalize.

Q Explain gradient Descent [GD] optimization technique & it variations [vanilla GD, stochastic GD, mini Batch GD]

→ Gradient Descent Optimization Technique

- Gradient Descent is the most common optimization algorithm in machine learning & deep learning.
- It is a first-order optimization algorithm. This means it only takes into account the first order derivation of the cost function when performing the updates on the parameters.

1. Vanilla GD:

It uses the entire dataset to compute the gradient of the cost function.

- Updates the model parameters after computing the average gradient over all training examples.
- Guarantees convergence to a global minimum for convex functions but can be slow for large datasets.
- Formula:

$$\theta = \theta - \alpha \frac{1}{m} \sum_{i=1}^{m} \nabla J(\theta, x_i, y_i)$$

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

2. Stochastic Gradient Descent :
Instead of computing the gradient the entire dataset it updates the parameters using only one random choosen training example at a time.
Formula :-

$$\theta = \theta - \alpha \nabla J(\theta, x_i y_i)$$

3. Mini-Batch Gradient Descent :-
Instead of using a single [SGD] or the entire data [Batch GD], it updates the parameters using a small subset [mini-batch] of training data.
This balances the efficiency of SGD and the stability of Batch GD.

7] Explain Momentum based GD and Nesterov ~~N~~ Accelerated GD ?

→ Momentum based GD :-

Gradient descent with momentum uses the momentum of the gradient for parameter optimization.

Parameters update in GD with momentum at iteration t, $\theta^{t-1} - v^t$ and $v^t = \beta v^{t-1} + \alpha \nabla L(\theta^{t-1})$

The term v^t is called momentum.

The parameter is reffered to as momentum

A typical value of the parameter is 0.9.

This method updates the parameters in the direction of the weighted average of the past gradients.

Nesterov Accelerated Momentum :-

• In some cases, the acceleration of momentum can be the search to miss or overshoot the minuma at the bottom of basins or valleys.

Nesterov momentum is an extension of momentum that involves calculating the decaying moving of the gradients of projected ~~proje~~ positions in the search of the gradients ~~of project~~ space rather than the actual positions themselves.

8] Why do you need regularization ? Explain underfitting & overfitting ?

→ Regularization is one of the most important concepts of Machine learning.
It is a technique to prevent the model from overfitting by adding extra information to it.
Sometimes ML model performs well with the training data but does not perform well with the test data.

Underfitting :-

Bias is the difference between the average prediction of our model & the correct value which we are trying to predict.
Model with high bias pay intention to the training data and overs simplifiers the model.
It always leads to high error on training and test data. This is called underfitting.

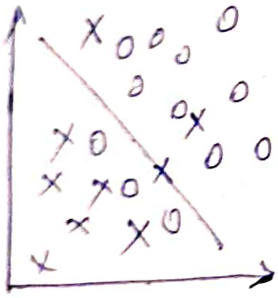
Overfitting :-

Variance is the variability of model prediction for given data point or a value which tells us spread of our data.

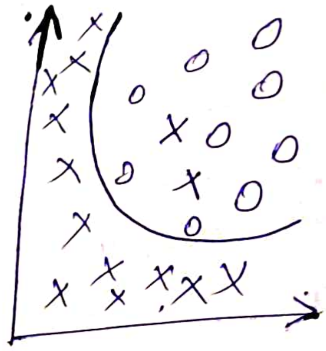
Model with high variance pays a lot of attention training data and does not generate well the data which it hasn't seen before.

As a result such models perform very well training data & does not generalize well the data which it hasn't seen before.

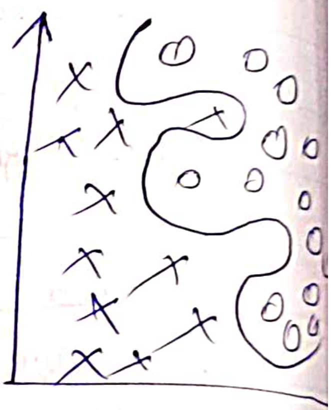
As a result, such models perform very well on training data but has high error rates on test data



Underfitting
[too simple to explain variance]



Appropriate-fitting.



Overfitting
[force fitting to good to be]

H. J. Thim Trust's

THEEM COLLEGE OF ENGINEERING

Date :

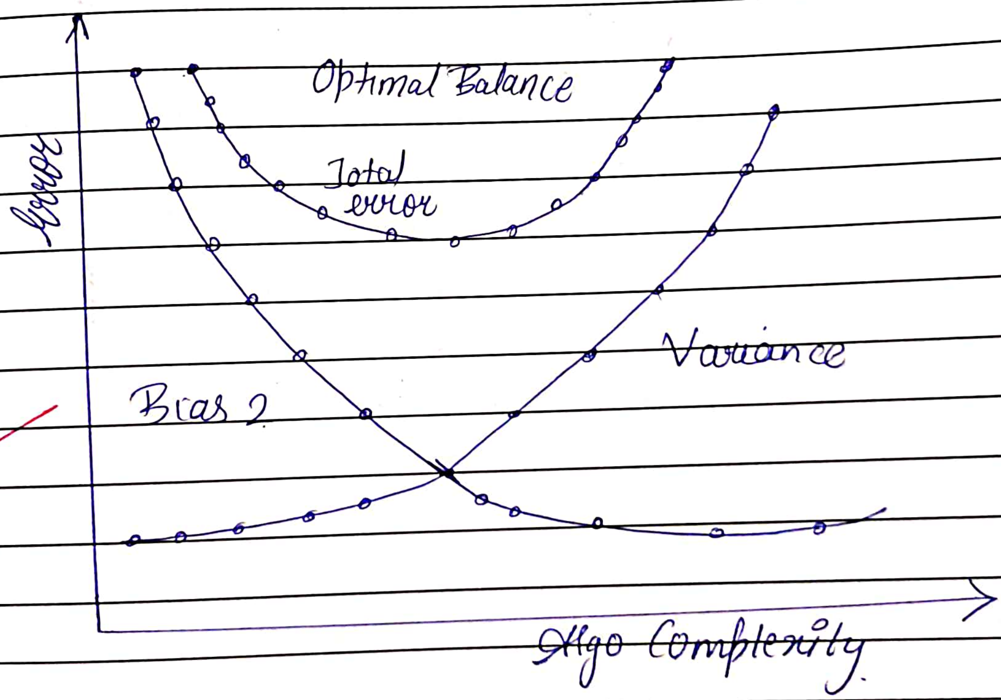
Q] What is Bias Variance trade-off?

→ 1. Bias variance trade-off is a term to describe the fact that we can reduce the variance by increasing the bias.

2. Good regularization techniques strive to simultaneously minimize the two source of error.

3. Hence achieving better generalization

4. To build a good model, we need to find a good balance between bias & variance such that it minimizes the total error.



10] Explain various regularization techniques such as weight, decay, L1 and L2 regularization drop out, early stop ping, batch normalization and data augmentation.

→ Various Regularization techniques:-

1. Weight Delay

- Real world data is complex & in order to solve complex problems we need complex solution.
- Having fewer parameters is only one way of preventing own model from getting over complex but it is actually a very limiting strategy.
- To prevent complexity we multiply the sum of squares with another smaller number called weight delay.

2. L1- Regularization -

A regression model that uses L1 Regularization techniques is called Lasso Regression. The Lasso Techniques ~~is~~ called Regression adds "absolute value of magnitude" of coefficient as penalty term to the loss function

$$\sum_{i=1}^{n} \left[y_i - \sum_{j=1}^{p} x_{ij} B_j \right]^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

3. L2 – Regularization

A model shows training loss gradually decreases but validation loss eventually goes up. In other words, the generalization curve shows that the model is overfitted to the data in the training set

Cost function: $\sum_{i=1}^{n} \left[y_i - \sum_{j=1}^{p} x_{ij} B_j \right]^2 + \lambda \sum_{j=1}^{p} B_j^2$

4. Drop Out Regularization.

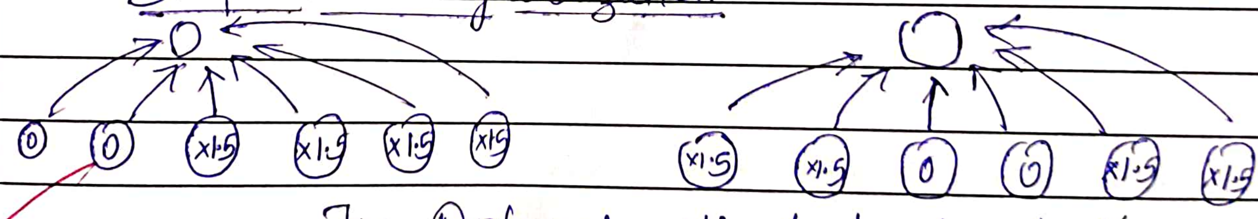


Fig Drop out with dropout rate 1/3

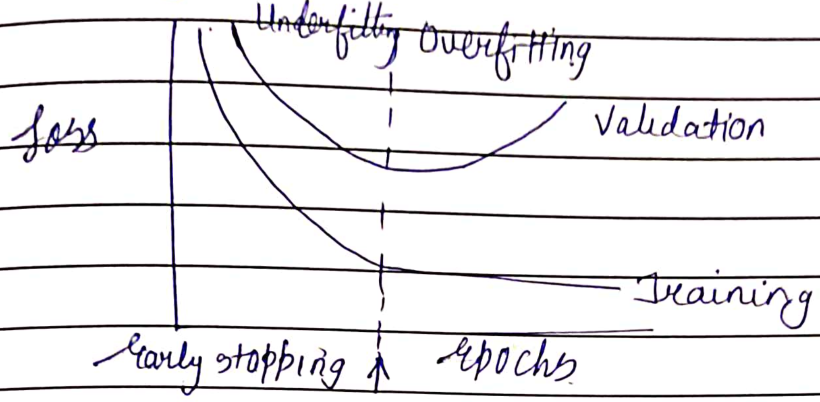
Two images represent dropout applied to a layer of shown at multiple training steps.

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

5. Early Stopping

A major challenge in training neural networks, is how long to train them.



6. Batch Normalization

It is a process to make neural networks faster & more stable through adding extra layers in a deep neural network.

It act similar to the data preprocessing steps mentioned earlier.

7. Data Augmentation

One of the reason why deep neural model overfits, underfits is lack of data.

Data can be augmentation by performing few transformation to the data like flipping, cropping adding noise to the data, etc.

Regularization also decreases a models variance.

H.J. Thim Trust's

Theem College of Engineering,

BoisarChilhar Road, Boisar (E), Tal - Palghar

ACADEMIC YEAR: 2024-25

CLASS: B.E Comp SEM: VIII

NAME: Siddhi P. Katkar

ROLL NO: 15

BATCH: B1

SUBJECT: Deep Learning

EXPERIMENT /ASSIGNMENT NO: 3

AIM: Assignment No. 3.

DATE OF PERFORMANCE: / /

DATE OF SUBMISSION: / /

PARAMETER	C	P	A	Total	Sign. With Date
MARKS OBTAINED	2	2	1	5	
MAX.MARKS	2	2	1	5	10/9

DL

THEEM COLLEGE OF ENGINEERING
H. J. Thim Trust's

Assignment No. 3

Date :

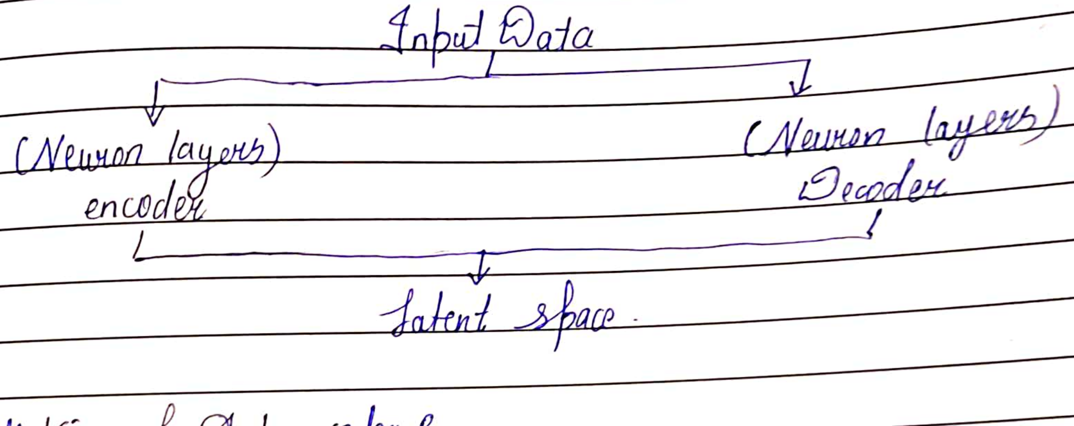
1] Draw and explain Architecture of Autoencoder.

→ An Autoencoder is a type of neural network used for unsupervised learning, mainly for dimensionality reduction, feature extraction and denoising data. It consists of two man components:

1. Encoder – Compresses input into a smaller representation (current state)
2. Decoder – Reconstructs the original input from the latent space.

- Input → [Encoder] → Latent space → [Decoder] → Output (Reconstructed inp)

Fig. Architecture Diagram.



- Working of Autoencoder :-

1. Encoder :
- Takes input x and compresses it to a lower dimensional representation (z).

- Uses activation function like Relu or sigmoid.
- form : $z = f(W_e x + b_e)$
  where :- W_e + b_e are encoder weights & biases f is an activation function.

2. Latent Space (Bottleneck) :
- Holds compressed essential features of inputs.
- Acts as a dimensionality reduction layer.

3. Decoder
- Reconstructs input from latent space z
- Tries to make output x as close as possible to input x.
- Mathematical form : $x = f(W_d z + b_d)$
  Where W_b, b_d are decoder weights & biases.

4. Loss function :
- Measures the difference between original input & reconstructed output.
- Uses mean squared error or Binary Cross entropy.

• Types of Autoencoder :-

1. Denoising Autoencoder - learns to remove noise from input
2. Sparse Encoder - Uses sparsity constraints to learn meaningf
3. Variational Autoencoder - Generates new data similar to input distribution

4. Convulutional Autoencoder – used for image processing with CNN layers

- Applications :–

- Image Compression
- Anomaly detection
- Data denoising
- Feature extraction

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

Q.2] Explain linear autoencoder.

→ A linear Autoencoder is a special type of autoencoder where both the encoder & decoder use linear transformations instead of non-linear activation function like ReLU or sigmoid. It essentially performs principal component Analysis (PCA) but in a neural network framework.

• Architecture :-

Input layer - Takes original high-dimensional input x.
encoder - Compresses the input to a lower-dimensional latent space.

$$z = W_e x + b_e$$

latent space - Stores compressed data.
Decoder - Reconstructs input from latent representation

$$x' = W_d z + b_d$$

Output - Reconstructed version x' of the original input.

• Advantages.
- Interpretable - Directly related to PCA.
- efficient & faster than non-linear autoencoder for dimensionality.
- No need for activation tuning - Simplifies training.

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

• Limitations :-

- Cannot model non-linear relationships in data.
- less expensive than deep autoencoders.

• Applications :

- Dimensionality reduction
-

- Preprocessing step for machine learning models.

H. J. Thim Trust's

THEEM COLLEGE OF ENGINEERING

Date :

3] Differentiate between under complete & over complete

→

Features	Undercomplete auto encoders	Over Complete
- Latent space Dimension	Smaller than input size $(z < x)$	Larger than input size $(z > x)$
- Purpose	efficient feature extraction, compression	Captures complex patterns, learns identity functions
- Risk	May lose some information	High risk of overfitting
- Regularization Needed?	Less required	Needs regularization
- Example	Dimensionality reduction (like PCA)	High-dimensional embedding anomaly detection.

4] Explain De-noising autoencoders

→ A Denoising autoencoders is a type of autoencoder designed to learn robust representations by removing noise from corrupted inputs.

- How it works :-

- Input Corruption - A noisy version of the input ~~into a~~ created by adding random noise.
- Encoder - compresses the noisy input into a latent representation.
- Decoder - Reconstructs the original (clean) input from the latent space.
- Loss function - Mean squared error (MSE) or Binary Cross entropy (BCE) is used to measure reconstruction accuracy

- Key Benefits :
- Improves robustness to noisy data
- Helps to feature extraction and denoising image/text
- Prevents overfitting by forcing the model to learn useful representation.

• Applications :

- Image Denoising
- Speech enhancement
- Pretraining deep networks.

5] Explain how autoencoders can be used for image compression.

→ Autoencoders can be used for Image Compression by learning a compact representation of a image & ~~ref~~ reconstructing it with minimal loss.

- Working :-

1. Encoder :
- Reduces the image dimensions by extracting key features.
- Converts input x into a compressed latent vector z

2. Latent space :
- Stores a lower-dimensional representation of the image.
- Acts as the compressed version of the original image.

3 Decoder :
- Reconstructs the image from z as closely as possible
- Outputs x', an approximation of x.

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

Applications :
- Medical Imaging - Storing MRI scans efficiently
- Face Recognition - Compressed embedding for fast recognition
- Video Compression - Reducing video storage bandwidth usage

- Ensure small variation in input donot drastically change latent representation
- Use case : Robust feature learning.

4. Variational Autoencoder (VAE) :
- Introduces ~~probistic~~ probablistic constraints by regularizing the latent space with a Gaussian distribution.
- Ensured meaningful continuous latent representation
- Use Case : Generative modeling, data synthesis

7] Discuss application of autoencoder for image denoising.

→ Autoencoders are widely used for image denoising where they learn to remove noise from corrupted images while preserving important details.

1. Training phase:
- The network is trained using noisy images as input and clean images as output.
- The autoencoder learns to map noisy images to their denoised versions.

2 Denoising Phase:
- When a new noisy image is given, the encoder extracts clean features and removes noise.
- The decoder reconstructs the noisy free image.

3. Applications:-
1. Medical Imagin (MRI, x-ray, CT Scans)
- Removes noise and artifacts from medical images to improve diagnosis accuracy.

2. Satellite Image Processing
- Enhances satellite and aerial images by reducing environmental noise.

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

3. Security and Surveillance
- Improves CCTV and low light images for facial recognition and object detection.

4. Old photo Restoration
- Removes specific noise and from old, degraded photography.

5. Document Image Enhancement
- Removes noise from scanned documents to improve text readability.

05/05

Tojgne 22/2/25

15

H.J. Thim Trust's

Theem College of Engineering,
BoisarChilhar Road, Boisar (E), Tal - Palghar

ACADEMIC YEAR: 2024-25 CLASS: B.E Comp SEM: VIII

NAME: Siddhi P. Katkar

ROLL NO: 15 BATCH: B1

SUBJECT: DL

EXPERIMENT /ASSIGNMENT NO: 4

AIM: Assignment No. 4

DATE OF PERFORMANCE: / /

DATE OF SUBMISSION: / /

PARAMETER	C	P	A	Total	Sign. With Date
MARKS OBTAINED	2	2	1	05	24/3/25
MAX.MARKS	4	4	2	10	

DL

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Assignment No. 4   Date :

1] Explain convolution operation with a suitable example.
→ Convolution is a mathematical operation used in image processing & DL. It helps a neural network detect patterns like edges, textures and shapes in an image.
In Convolutional Neural Network (CNNs), convolution is the core operation that helps extract important features from an image before classification.

Convolution is performed using two main components:
1. Input Image (Matrix): The image is represented as a matrix of pixel values.
2. Kernel (Filter): A small matrix (e.g 3x3 or 5x5) that slides over the image & performs element-wise multiplication and summation to create a new matrix called a feature map.

Step by step example –
E.g: Edge Detection Using a 3x3 Kernel

Consider a 5x5 grayscale image (only one channel) & a 3x3 filter (Kernel):

→

Input image (5x5)

3x3 Kernel (edge Detection)

Feature Map (3x3 output)

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \rightarrow \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix}$$

Diagramatic Representation

H. J. Thim Trust's

THEEM COLLEGE OF ENGINEERING

Date :

Step 1 : Input Image (5x5 Matrix)

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 6 & 7 & 8 & 9 \\ 9 & 10 & 11 & 12 & 13 \\ 13 & 14 & 15 & 16 & 17 \\ 17 & 18 & 19 & 20 & 21 \end{bmatrix}$$

Step 2 : Kernel (3x3) filter for edge Detection.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Step 3 : Applying Convolution

- The Kernel slides over the image, performing element-wise multiplication & summing the result.
- This enhances certain features (like edges) while reducing information

For the top-left corner (3 x 3 region of the image):

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 9 & 10 & 11 \end{bmatrix}$$

Multiply element-wise with the Kernel:

$(1\times-1)+(2\times-1)+(3\times-1)+(5\times-1)+(6\times8)$
$+(7\times-1)+(9\times-1)+(10\times-1)+(11\times-1)$

$= -1-2-3-5+48-7-9-10-11=0$

This value is stored in feature map at the corresponding position.

Step 4: Feature Map (Output After Convolution)
After sliding the filter across the entire image, we get a smaller output matrix.
Eg,
I/P is 5x5 & the Kernel is 3x3, the output will be 3x3 as padding is not used:

$$\begin{bmatrix} 0 & 4 & 8 \\ 12 & 16 & 20 \\ 24 & 28 & 32 \end{bmatrix}$$

This matrix highlights edges in the image.

2] Explain the following techniques with respect to CNN:
i) Pooling

→ Pooling is a down-sampling technique used in CNNs to reduce the size of feature maps while retaining important information.

Why use Pooling?
- Reduces computational cost.
- Controls overfitting by reducing the number of parameters.
- Makes the network more ~~to~~ robust to variations like rotation & small changes in position.

* Types of Pooling

1. Max Pooling
Selects the maximum value from the given region.
Helps detect strongest features like edges & textures.

2. Average Pooling
Takes average value from given region

Preserves more background details but is less effective than Max Pooling for feature extraction.

3. Global Pooling
Uses one pooling operation over the entire feature map to get a single value per feature map.
Reduces this feature map to a single number, often used before fully connected layers.

ii) Stride
Stride is the step size that the filter (Kernel) moves over the image during convolution or pooling.
If stride = 1, the filter moves one pixel at a time (default).
If stride = 2, the filter moves two pixels, reducing the o/p size.
Higher stride → smaller output feature map.

Example of Stride = 1 vs Stride = 2

With Stride = 1 :
Input Image : 5x5

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

Kernel : 3 x 3
O/P feature Map : 3 x 3 (moves 1 pixel at a time)

With Stride = 2 :
Input Image : 5 x 5
Kernel : 3 x 3
O/P feature Map : 2 x 2

Why use Stride ?
1. Reduces computation by decreasing feature map size.
2. Helps control the amount of detail captured
3. Stride greater than 1 helps speed up CNNs by reducing the data size.

iii) Padding
Padding adds extra pixels (usually zeros) around the input image before applying convolution.
Types of Padding.

1. Valid Padding (No Padding)
No extra pixels are added.
The O/P size becomes smaller than the i/p size.

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

E.g.

Input : 5 x 5
Kernel : 3 x 3
Output : 3 x 3 (reduced size).

2. Same Padding (Zero Padding)
Adds zeros ~~at~~ around the image to keep the same size after convolution.
Commonly used in CNNs to maintain the feature map size.
E.g

Input : 5 x 5
Kernel : 3 x 3
Output : 5 x 5 (same size as input)

Why use Padding.

Prevents loss of ~~infro~~ important edge features.
Preserves spatial dimensions after convolution
Helps apply deeper networks without shrinking image size too fast.

3] Explain CNN architecture in detail

→ A CNN typically consists of following layers :-

1. Input Layer

The i/p to a CNN is an image, represented as a matrix of pixel values.

E.g: A 28x28 grayscal image has pixel values between 0-255. If it's RGB, the image has 3 color channels (Red, Gree, Blue)

2. Convolutional Layer

The most important layer in CNN.

Uses filters (Kernels) to scan the image & extract important features (edges, shapes, etc)

Each filter moves over the image, performing a dot product between filter values & pixel values.

Produces a feature map that highlights different aspects of image.

E.g:

Input : 5x5 image

Filter : 3x3 matrix

Output : 3x3 feature map

3. Activation Function (ReLU - Rectified Linear Unit)

Adds non-linearity to the network.

Converts all negative values to zero (ReLU = max(0, x)).
Helps the CNN learn complex patterns efficiently.
Eg
Before ReLU : [-2, 3, -5, 7]
After ReLU : [0, 3, 0, 7]

4. Pooling Layer
Reduces size of feature maps while keeping important information.
Makes CNN faster & less prone to overfitting.
Two types.
- Max Pooling - Takes maximum value from each region.
- Average Pooling - Takes the average value from each region.

5. Fully Connected Layer
Flattens the feature map into a single vector
Passes it to a fully connected layer.
Each neuron learns a different combination of extracted features to classify image.
Eg:
Cat → 0.85
Dog → 0.15
The network predicts Cat (highest probability)

6. Output Layer (Softmax / Sigmoid Function)
Converts final values into probabilities for classification.
Softmax is used for multi-class classification (e.g digits 0-9)
Sigmoid is used for binary classification (e.g cat vs dog)
eg :-
Softmax Output for 3 Classes.
Dog : 0.1.
Cat : 0.8
Elephant : 0.1

The CNN classifies image as a Cat (highest probability = 0.8)

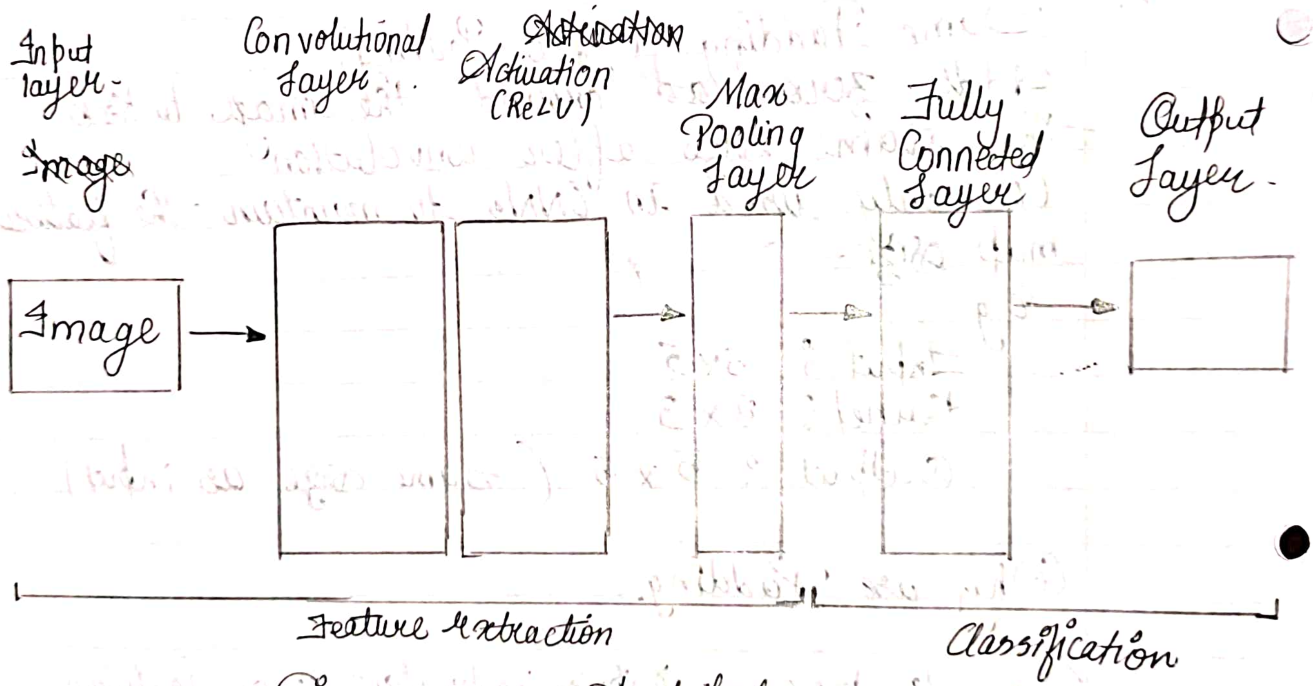


Diagram : CNN Architecture

Input

4] Explain architecture of AlexNet.

→ AlexNet is a deep convolutional neural network that won the ImageNet Large Scale Visual Recognition Challenge in 2012. The network consists of 8 layer : five convolutional layers followed by three fully connected layers.

1. Input layer
The input to AlexNet is a RGB image of size 227 x 227 x 3
The image is preprocessed by substracting the mean intensity value computed over the dataset.
The input is fed into first convolutional layer.

2. Convolutional Layers.
AlexNet uses five convolutional layer. Each layer applies a set of learnable filters to detect pattern in i/P image.

(a) Conv1 (First Convolutional layer)
Filters : 96 filters of 11 x 11 x 3
Stride : 4   Padding : 0
Activation Fun: ReLU
O/P Feature Map size : 55x 55x96.
Additional Processing : Max Pooling reduces the feature map size to 27x27x96

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

(b) Conv 2
Filters : 256 filters of 5x5x48
Stride : 1 Padding : 2
Activation Fun : ReLU
O/P feature map Size : 27x27x256
Additional processing : Max pooling reduces the size to 13x13x256.

(c) Conv 3
Filters : 384 filters of 3x3x256
Stride : 1 Padding : 1
Activation Function : ReLU
O/P feature Map Size : 13x13x384

(d) Conv 4
Filters : 384 filters of 3x3x192
Stride : 1 Padding : 1
Activation Fun : ReLU
O/P Feature Map Size : 13x13x384

(e) Conv 5
Filters : 256 filters of 3x3x192
Stride : 1 Padding : 1 Activation Fun : ReLU
O/P feature Map Size : 13x13x256
Additional Processing : max pooling reduces size to 6x6x256

3. Fully Connected Layers

Feature maps obtained from last convolutional layer are flattened & fed into fully connected layer.

(a) First fully Connected layer - Neurons - 4096 - ReLU

(b) Second fully connected layer - Neurons - 4096 - ReLU

(c) Third fully connected layer - Neurons - 1000 - Softmax.

4. Additional feature in AlexNet

ReLU Activation
Dropout Regularization
Overlapping Max Pooling

5. Output layer.

Final layer applies Softmax activation, it converts final scores into probabilities for each of 1000 classes. The class with highest probability is selected as the predicted label.

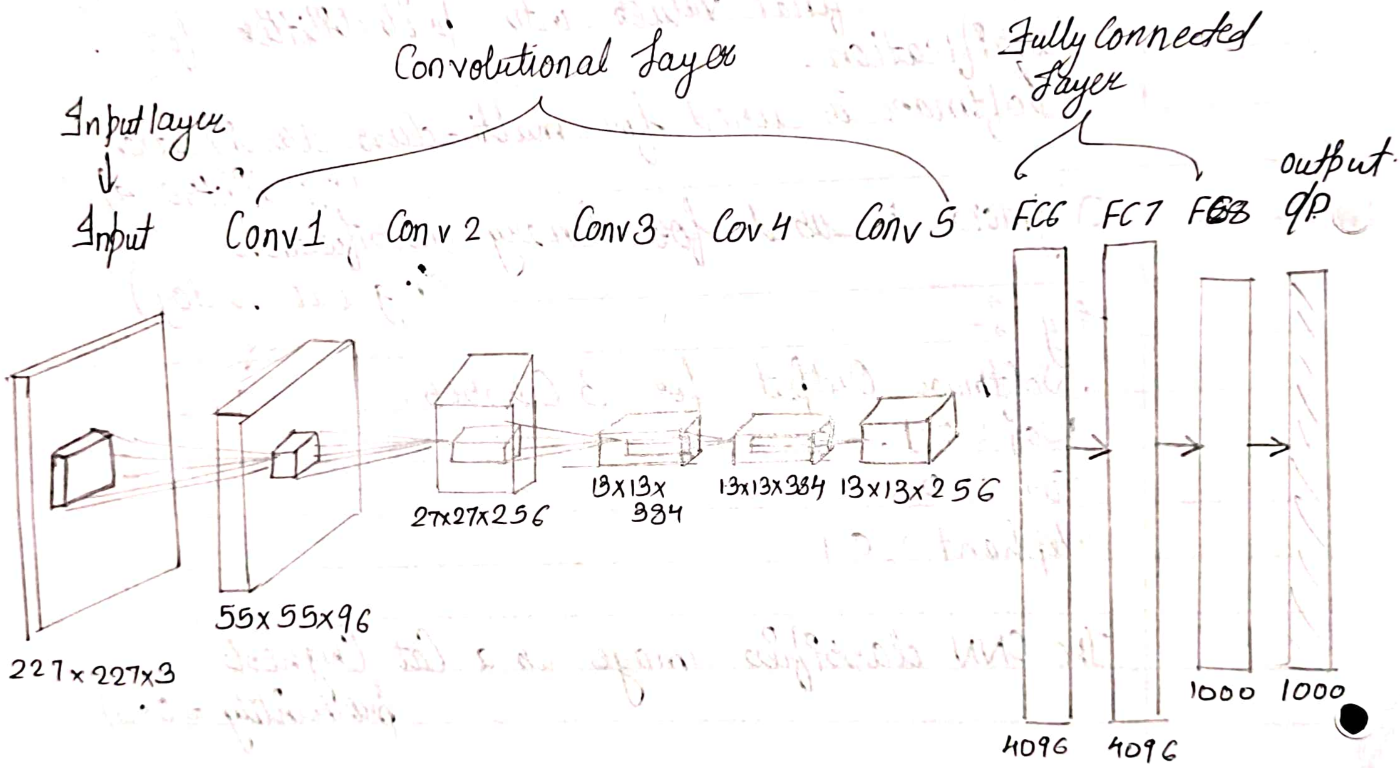


Diagram: Architecture of AlexNet

H. J. Thim Trust's

THEEM COLLEGE OF ENGINEERING

Date :

5] Differentiate between Fully Connected Network and CNN

→

	Fully Connected Network (FCN)	Convolutional Neural Network (CNN)
1. Structure	All neurons are connected to each other.	Uses convolutional layers with filters.
2. Data Type	Works well with structured/tabular data.	Best for image & spatial data.
3. Feature Extraction	Requires manual feature extraction.	Automatically extracts features.
4. Parameter count	Large no. of parameters (weights)	Fewer parameters due to shared filters.
5. Computational Cost	High, due to dense connections.	Lower, due to weight sharing.
6. Spatial Inf.	Does not preserve spatial relationships.	Preserves spatial features in data.
7. Overfitting	More prone to overfitting.	Less prone due to parameter sharing.
8. Training time	Longer due to large parameter count.	Faster training with fewer parameters.
9. Use Case	Used for structured data, like fraud detection.	Used for image processing, obj. detection.
10. Flexibility	Less flexible for unstructured data.	More adaptable to visual & complex data.

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

6] Differentiate between LeNet and AlexNet

→

	LeNet (1990 s)	AlexNet (2012)
1. Developed by	Yann LeCun	Alex Krizhevsky
2. Purpose	Handwritten digit recognition	Large scale image classification
3. Input size	32x32x1 (Grayscale)	227x227x3 (RGB)
4. Depth (Layers)	7 layers	8 layers
5. Filter used	Small (5x5)	Large (11x11, 5x5, 3x3)
6. Activation	Sigmoid / Tanh	ReLU (Faster training)
7. Pooling	Average Pooling	Max Pooling
8. Dropout	No dropout	Uses dropout (Reduces overfitting)
9. Computational Cost	Low (Simple network)	High (Deep n/w, needs GPUs)
10. GPU usage	No GPU required.	Requires GPU for training
11. Performance	Good for small datasets	Excellent for large datasets.

H.J. Thim Trust's

Theem College of Engineering,
BoisarChilhar Road, Boisar (E), Tal - Palghar

ADEMIC YEAR: 2024-25 CLASS: B.E Comp SEM: VIII

AME: Siddhi Katkar

OLL NO: 15 BATCH: B1

UBJECT: DL

XPERIMENT /ASSIGNMENT NO: 5

IM: Assignment No. 5

ATE OF PERFORMANCE: / /

ATE OF SUBMISSION: / /

PARAMETER	C	P	A	Total	Sign. With Date
MARKS OBTAINED	2	2	1	05	Rojane 24/3/25
MAX.MARKS	4	4	2	10	

DL

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Assignment No. 5 Date :

1] What are the different types of RNN architectures?
→ Recurrent Neural Networks are a class of neural networks designed for processing sequential data (e.g. time series, Natural language, Speech recognition). Unlike traditional feedforward networks, RNNs maintain memory of previous input using recurrent connections.
RNN architecture are categorized based on how information flows between layers & across time steps.
Types of RNN Architectures are :-

1. Simple (Vanilla) RNN
The basic form of RNN, where each neuron maintains a hidden state that carries information from previous time steps.
Uses a feedback loop to process sequential data.
Application:
Basic sequential data processing like stock price prediction.

2. Long Short-Term Memory
A specialized RNN architecture designed to overcome the vanishing gradient problem.

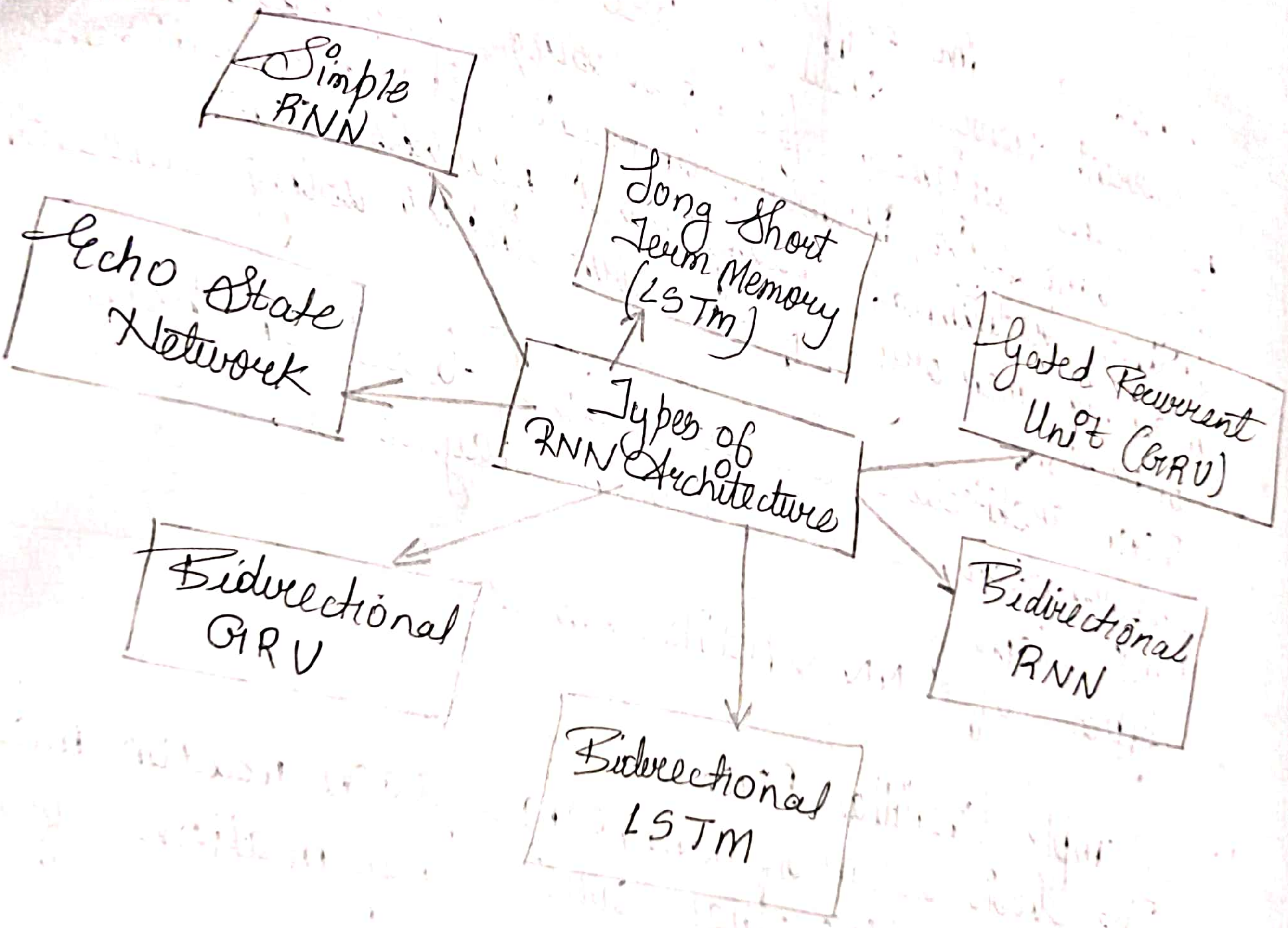


Diagram: Types of RNN Architecture

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

Application:
Speech recognition (Google Assistant, Siri)
Machine translation (Google Translate)

3. Gated Recurrent Unit (GRU)
A simplified version of LSTM with fewer parameters.
Uses only two gates:
Reset Gate: Controls how much past information is forgotten.
Update Gate: Determines how much past information is retained.
Application:
Real-time applications
Time-series prediction

4. Bidirectional RNN (Bi-RNN)
Consists of two RNNs:
One processes data forward (past to future)
One processes data backward (future to past)
Application:
Speed recognition
Named Entity Recognition.

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

5. ~~Function~~ Bidirectional LSTM (Bi-LSTM)
A combination of directional processing & LSTM memory units.
Captures both past & future dependencies in long sequences.
Application:
Automatic text Summarization
Language translation models.

6. Bidirectional GRU.
Similar to Bi-LSTM but uses GRU units for efficiency.
Advantage:
Faster computation than Bi-LSTM but with slightly lower accuracy in some tasks.
Application:
Sentiment analysis
Speech emotion recognition

7. Echo State Network (ESN)
A variant of RNN where only output weights are trained while the recurrent connections remain fixed.
Application: Time series forecasting.
Dynamical system modeling.

3) What is GRU (Gated Recurrent Unit)?
A Gated Recurrent Unit is a type of RNN architecture designed to handle sequential data efficiently. GRU solves the vanishing gradient problem in standard RNNs while being computationally more efficient than LSTMs.

Architecture of GRU :-
GRU consists of two gates
1. Update Gate (z_t)
Controls how much of the past information should be retained in the current hidden state. Functions similarly to the forget gate & input gate in LSTMs (combined)
2. Reset Gate (r_t)
Controls how much of the past information should be forgotten before calculating the new candidate hidden state.

Mathematical Equations.
Let,
X_t = Input at time t
h_t = Hidden state at time t
W, U, b = Weight matrices and biases
σ = Sigmoid activation function

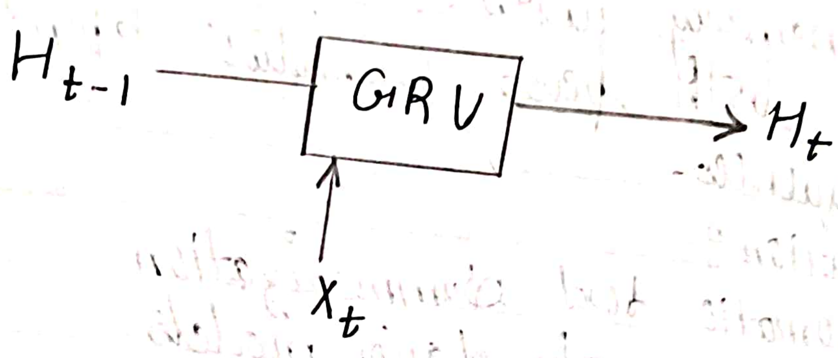


Diagram: GRU cell.

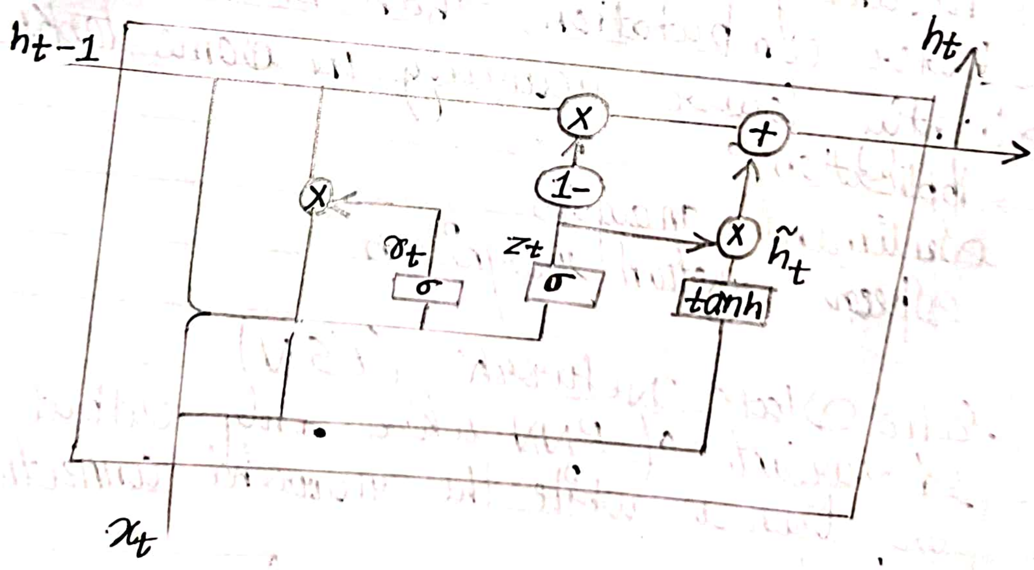


Diagram: GRU (Gated Recurrent Unit)

tanh = Hyperbolic tangent activation function.

The GRU updates are computed as follows:

1. Update Gate (z_t)
Controls how much of the past information should be retained in the current

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$

Determines how much of the past hidden state should be carried forward.

2. Reset Gate

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$

Determines how much of the previous hidden state should be reset.

3. Candidate Hidden State

$$\tilde{h}_t = \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$$

Computes the new candidate activation.

4. Final hidden State

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

A weighted sum of the old hidden state & the newly computed candidate hidden state.

H. J. Thim Trust's

THEEM COLLEGE OF ENGINEERING

Date :

Key features :-

1. Computationally efficient
2. Faster training
3. Performs well on small datasets
4. Solves vanishing gradient problem.

3] Define LSTM. How does LSTM solve the vanishing gradient problem?

→ A Long Short-Term Memory is a special type of Recurrent Neural Network (RNN) designed to handle sequential data like text, speech and time series data.

The vanishing gradient problem occurs in standard RNNs when gradients become very small during back-propagation, preventing the network from learning long-term dependencies.

LSTM solves this by:

1. Using the Cell State (Ct)

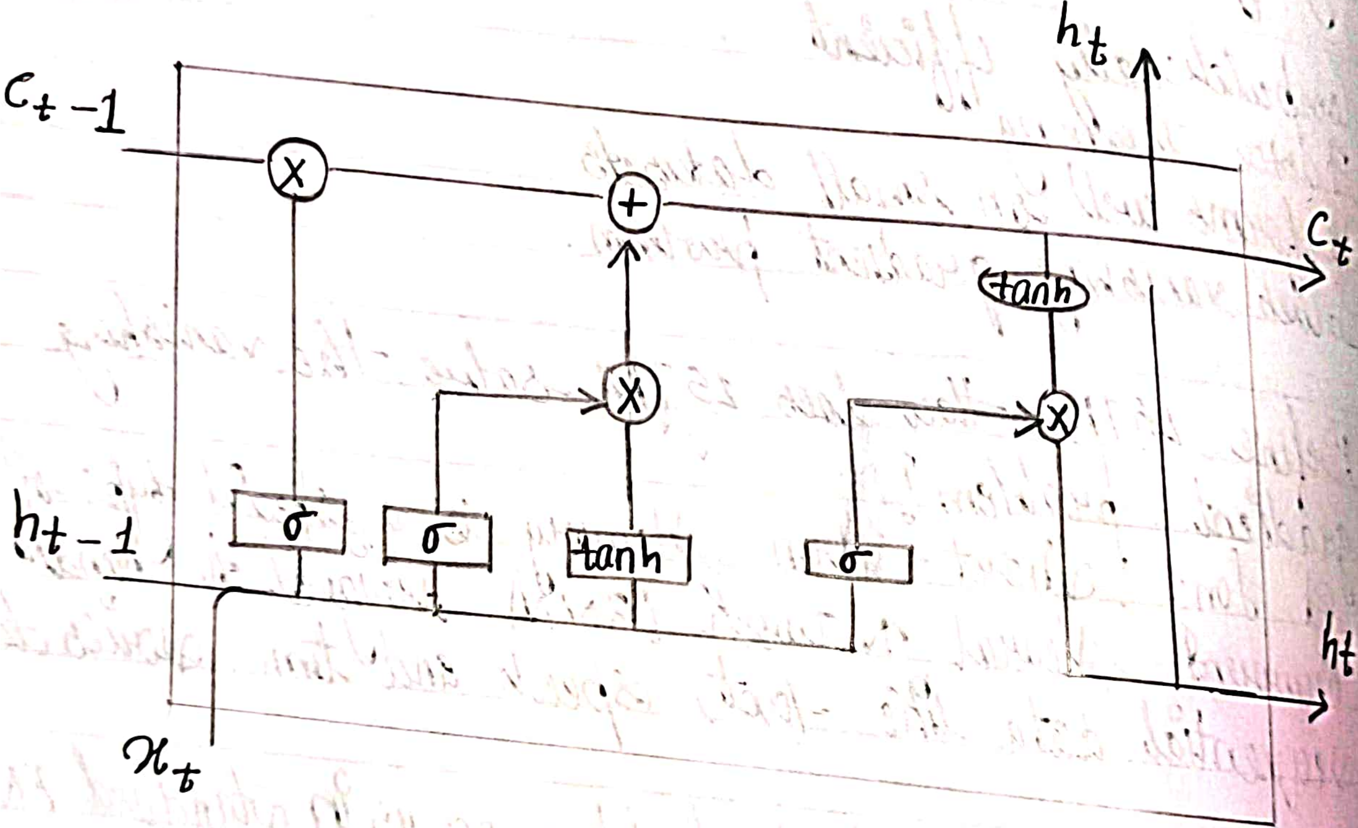
The cell state allows information to flow without getting multiplied repeatedly, preventing it from shring to zero.

2. Using Gates
The Forget Gate removes unimportant information
The Input Gate carefully selects new information to add.
This controlled flow ensures that important gradients don't disappear.

3. Using the Identity Function for Memory
The cell state uses a direct connection (addition instead of multiplication), which helps in keeping gradient stable over time.

~~Wr.~~ Use of LSTM

1. Speech Recognition (Google Assistant, Siri)
2. Machine Translation (Google Translate)
3. Chatbots (AI Conversations)
4. Stock Market Prediction (Time Series forecasting)



LSTM
(Long-Short Term Memory)

5] What is the exploding gradient problem?

→ The exploding gradient problem occurs when the gradients become too large during training causing the model's parameters to change drastically. This leads to unstable training and prevents the neural network from learning properly.

It mainly occurs in deep neural networks & RNNs especially when working with long sequences.

~~Dur~~ Why does Exploding Gradient Problem Occur?

During backgrou propogation (training phase) the gradients are updated are too large, the gradient values increase exponentially instead of remaining balanced.

Mathematically if we keep multiplying large values in deep networks we get:

$$\text{Gradient} = w_1 \cdot w_2 \cdot w_3 \cdots w_n$$

If $w_i > 1$, the gradients grow exponentially, causing instability.

How to fix the Exploding Gradient Problem?

1. Gradient Clipping
Limit the size of gradients to a fixed range (e.g. [-5, 5]) to prevent from becoming too large.
E.g :-
If $|| \nabla W || >$ threshold scale down gradients. ~~implemented~~
implemented in TensorFlow / PyTorch easily.

2. Use Proper Weight Initialization
Initialize weights using Xavier or He initialization to prevent extreme values at the start.

3 Lower the Learning Rate
Using a smaller learning rate prevents drastic weight update.

4. Use LSTM / GRU instead of Vanilla RNNs
LSTMs and GRUs control the flow of gradients, reducing instability

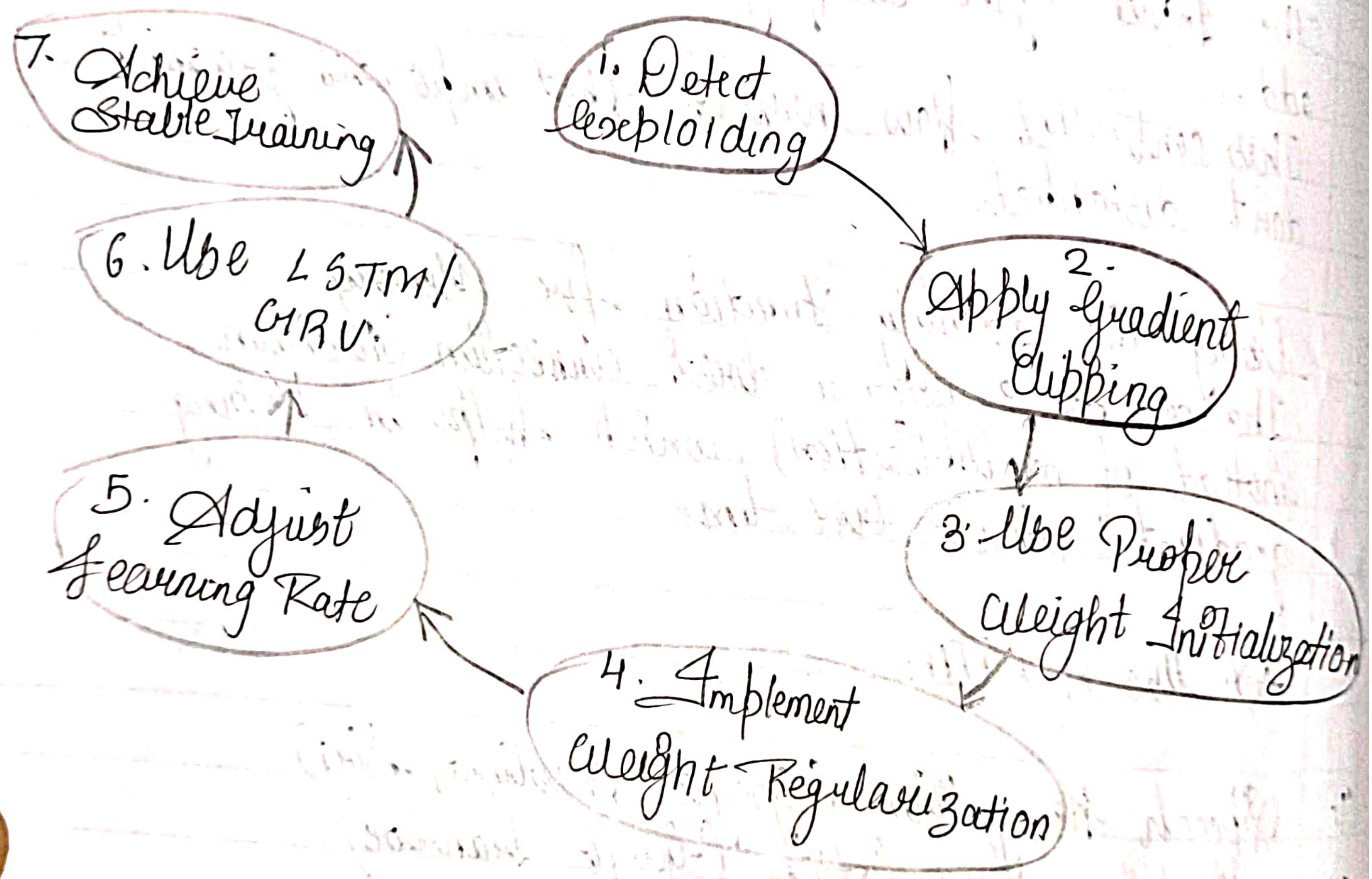


Diagram:- How to fix exploding Gradient Problem.

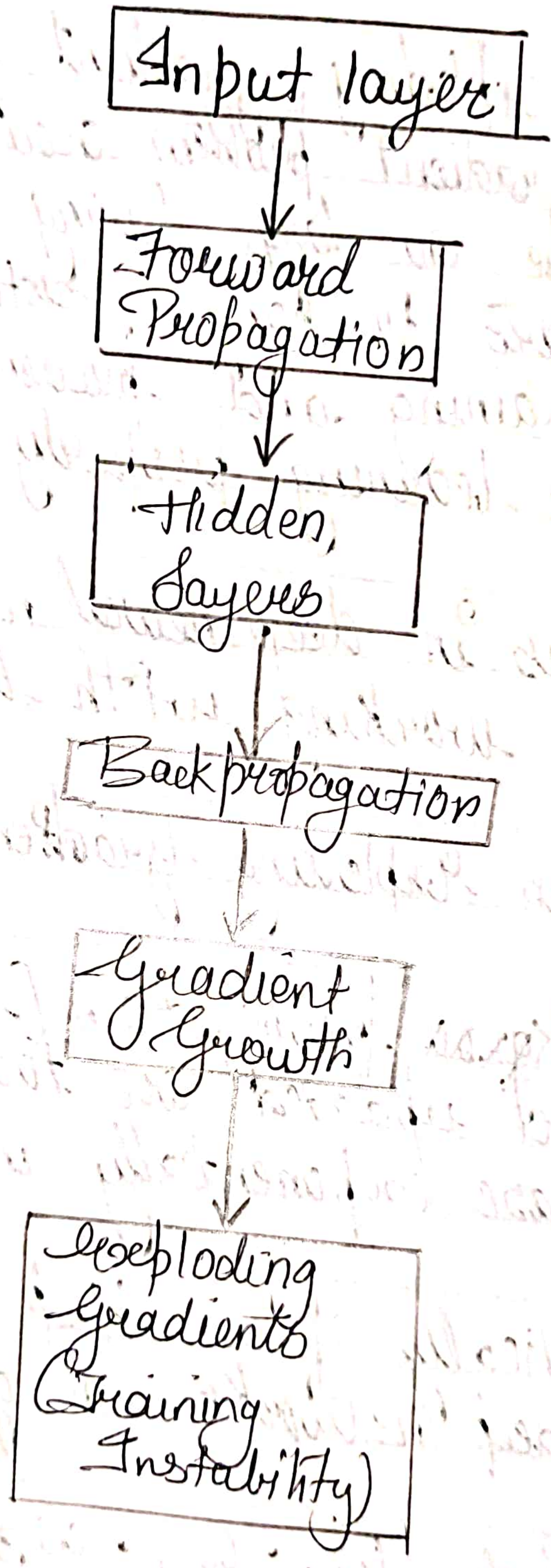


Diagram :- What is exploding Gradient Problem.

4) State & Explain Recurrent Neural Network (RNN)

→ A Recurrent Neural Network is a type of AI that helps computers understand & process sequential data (data that comes in order, like sentences, music or time-series data)

Imagine you are reading a book. To understand a sentence you don't just look at a current word, you also remember words that come before. RNN works the same way. It remembers past info. while processing new data

1. Takes Input - The network receives one word at a time

2. Remembers Previous Information - It stores some memory from the previous step.

3. Gives Output - Based on new input + past memory, it makes a prediction or decision.

4. Repeats for Next Step - The cycle continues for the next input.

This memory is what makes RNN special. Unlike normal neural networks, which process data independendly, RNN connects past & present information.

- Problems with RNN.

1. Forgets long-term information.
2. Slow training
3. Vanishing gradient.

To solve this problems, advanced version like LSTM & GRU were created. They have better memory management.

- Voice Applications-

1. Chatbots.
2. Text Prediction
3. Voice Assistant
4. Music & Speech Generation

and explain its types.

5] Explain Long Short Term memory (LSTM)

→ LSTM is a special type of RNN capable of learning-term dependencies by introducing a memory cell & gating mechanisms.

Key components :

Cell State :

- Forget Gate :
  Decides which information should be removed from cell state.
- Input Gate
  Determines which new info. is added to the cell state.
- Output Gate
  Controls what information from cell state is output at current time step.

Working :-

At each time step, the LSTM cell computes

- How much of previous cell state to retain (forget gate)
- What new information to add (input gate)
- And what to output (output gate)

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

These gates use sigmoid activations & element-wise multiplication to control the flow of information allowing LSTM to mitigate both vanishing & exploding gradients.

• Advantages of LSTM.

1. Solves the vanishing gradient problem.
2. Selective memory retention
3. Effective in sequential data tasks.

Applications.

1. Machine Translation
2. Text Generation
3. Stock Market Prediction.
4. Speech Recognition.

H.J. Thim Trust's

Theem College of Engineering,
BoisarChilhar Road, Boisar (E), Tal - Palghar

ACADEMIC YEAR: 2024-25 CLASS: B.E Comp. SEM: VIII

NAME: Siddhi P. Katkar

ROLL NO: 15 BATCH: B1

SUBJECT: DL

EXPERIMENT /ASSIGNMENT NO: 6

AIM: Assignment NO. 6

DATE OF PERFORMANCE: / /

DATE OF SUBMISSION: / /

PARAMETER	C	P	A	Total	Sign. With Date
MARKS OBTAINED	02	01	01	04	24/3/25
MAX.MARKS	4	4	2	10	

DL

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Assignment No. 5

Date :

1] What are generative adversarial network (GANs), and explain their types?

→ Generative Adversarial Network (GANs) are a class of deep learning models used for generative new data that resembles a given dataset.

They consist of two neural networks:

Generator (G): Generates fake data that looks like real data.

Discriminator (D): Tries to classify whether the input data is real or fake.

GANs are trained using an adversial process where the Generator improves in generating realistic data while the Discriminator improves in detecting fakes.

Types of GANs.

1. Vanilla GAN

The basic form of GAN introduced by Ian Goodfellow in 2014.

Uses a minimax optimization function, where the Generator tries to minimize the Discriminator's ability to classify real vs fake data.

Uses fully connected neural network for both Generator and Discriminator.

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

Challenges :
Mode collapse : The Generator produces only a few types of samples instead of diverse outputs.
Training instability : Difficult to balance training between Generator and Discriminator

2. Deep Convolutional GAN (DCGAN)
Uses Convolutional Neural Network (CNNs) instead of fully connected layers to improve image generation.
Employs techniques such as :
Batch Normalization - Stabilizes training.
Leaky ReLU Activation - Prevents dead neurons.
Transposed Convolutions - Improves Generator performance

Applications - Image generation and enhancement

3. Conditional GAN (cGAN)
Conditioned on additional input (e.g., class labels)
Both Generator & Discriminator receive extra info. that helps in controlled image generation.
Applications:
Generating images based on text descriptions.
Medical imaging (e.g generating disease specific MRI scans)

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

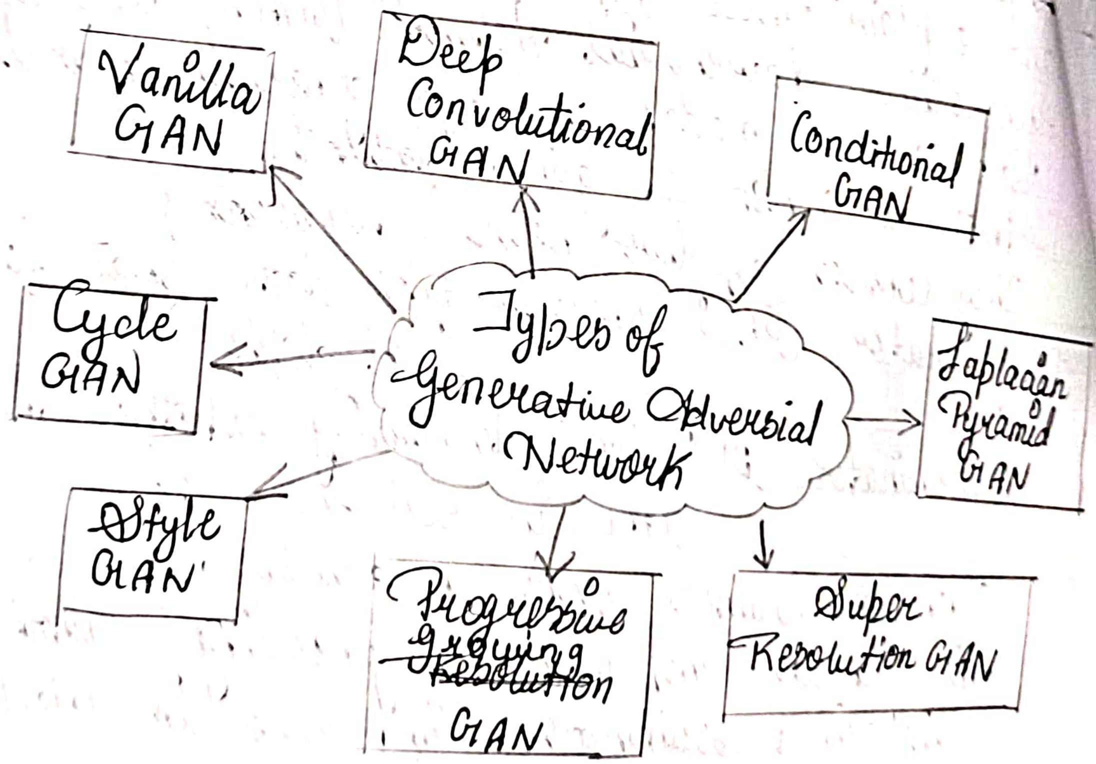
4. Laplacian Pyramid GAN (LAPGAN)
LAPGAN is designed to generate ultra high quality images by leveraging a multi-resolution approach.
Working of LAPGAN:
Uses multiple generator discriminator pairs at different levels of the Laplacian pyrid.
— Images are first downsampled at each layer of the pyramid and upscaled again using Conditional GANs.
This process allows the images to gradually refine details, reducing noise & improving clarity.

5. Super Resolution GAN (SRGAN)
Used for enhaning image resolution
Uses perceptual loss to preserve fine details.
Application:
— Image upscaling in medical imaging & satellite imagery.

6. Progressive Growing GAN (PGGAN)
Trains progressively from low to high resolution
Used for high-quality image sythesis
Applications:
— Realistic human face generation.

6. StyleGAN
Introduces style based generation, where different aspects of an image (e.g hair, age, expression) can be controlled independently.
Application :
- High Resolution image synthesis (e.g AI-generated human faces)

7. CycleGAN
Used for image to image translation without paired training data.
Converts images from one domain to another.
Application :-
Style transfer in images
Converting paintings into photographs


Types of Generative Adversial Network
Vanilla GAN
Deep Convolutional GAN
Conditional GAN
Cycle GAN
Laplacian Pyramid GAN
Style GAN
Progressive Growing ~~Resolution~~ GAN
Super Resolution GAN